ETP Using GA & ACO

(EA Project)

Team Members

1.	Mahmoud Mohamed Ahmed Sayed	20210876
2.	Adel Ali Ibrahim Ali	20210481
3.	Sayed Yosry El-Sayed Ragab	20220217
4.	Mohamed Ahmed Mohamed Abu El-Hassan	20220380
5.	Hussein Mohamed Hussein	20220139

Project Overview

Instructor: Dr. Amr.s Ghoneim.

Problem Definition: The Exam Timetabling Problem involves scheduling examinations while respecting numerous constraints:

Hard Constraints:

- No student can attend two exams simultaneously
- Room capacity must not be exceeded
- Exams requiring multiple timeslots must use consecutive slots
- Rooms cannot be double-booked

Soft Constraints:

- Students should not have multiple exams on the same day
- Difficult exams should be well-distributed
- Weekend scheduling should be minimized
- Exams should be spread evenly across the available period

Project description: This project implements and compares two metaheuristic approaches—Genetic Algorithm (GA) and Ant Colony Optimization (ACO)—to solve the complex problem of exam timetabling in educational institutions. The system optimizes exam schedules by minimizing student conflicts, balancing exam difficulty distribution, and ensuring efficient use of available resources.

Github Repository Link: https://github.com/MahmoudDiab152/housing-prices-ensemble-methods.git

ETP Optimization using GA Algorithm and ACO

1. Introduction

The Exam Timetabling Problem (ETP) is a widely known NP-hard problem that involves assigning a set of exams to a limited number of timeslots and rooms while satisfying a range of constraints.
This project uses a hybrid approach combining Genetic Algorithms (GA) and Ant Colony Optimization (ACO) to solve the problem more efficiently than traditional heuristic methods.

2. Problem Definition

Given a list of exams, students, and rooms, assign each exam a time slot and room such that:
- No student has overlapping exams.
- Room capacities are not exceeded.
- Time slots are used efficiently.
- Student workloads are balanced.

3. Objectives

- Minimize the number of student exam conflicts.
- Maximize the utilization of rooms and time slots.
- Distribute student exams uniformly over the exam period.
- Minimize the number of consecutive exams for students.

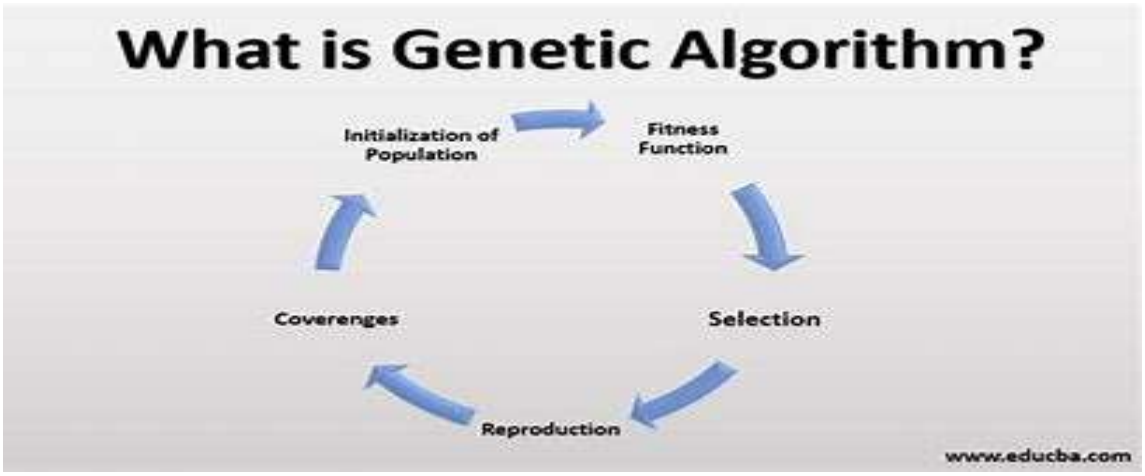
4. Constraints

- Hard Constraints:
  - No student has two exams at the same time.
  - Room capacity must not be exceeded.
- Soft Constraints:
  - Minimize number of students having more than one exam per day.
  - Spread exams evenly for students across available days.
  - Prioritize early allocation of high-enrollment exams.

Genetic Algorithm (GA) Overview

How it works:

1. **Start with random solutions.**
2. **Pick the best ones.**
3. **Mix them** to make new solutions.
4. **Sometimes change them a little.**
5. **Repeat** until you find a great solution.



GA is used to generate an initial population and evolve it:

Chromosome:

One timetable encoding all exams to time slots and rooms.

Fitness Function:

Penalizes student conflicts and rewards efficient spacing.

Operators:

Selection: Tournament selection.

Crossover: One-point and two-point crossover for mixing

schedules.

Mutation: Randomly reallocates an exam to a different slot.

Selection:

Uses **Tournament Selection**, where a subset of the population is randomly sampled, and the best among them is chosen for reproduction.

Crossover:

Implements **Two-Point Crossover**, swapping segments of two parent schedules to produce new offspring

Mutation:

Three mutation strategies are applied randomly:

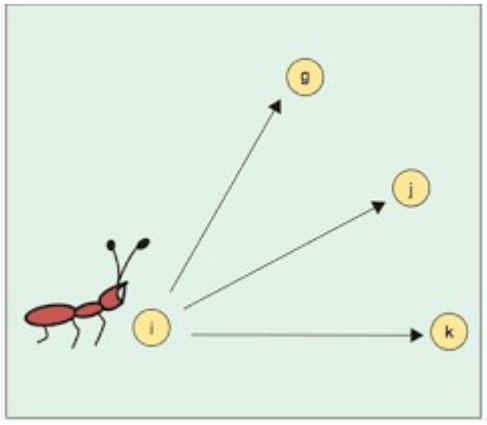
- Timeslot mutation (shift, add, remove)
- Room mutation (change/add/remove)
- Combined mutation (timeslot + room)

Ant Colony Optimization (ACO) Overview

How it works:

1. **Ants walk around randomly** looking for food.
2. When they **find food**, they go back and leave a **smell trail (pheromone)**.
3. Other ants follow strong smell paths (shorter paths get more smell).
4. Over time, **best paths stay**, bad ones disappear.

ACO simulates ants exploring paths between nodes (exams, slots):



Ants incrementally build timetables.

Pheromones encourage selection of good paths (exam-room-time combinations).

Global and local pheromone updates reinforce or degrade path likelihood.

Probabilistic selection allows exploration and exploitation.

Pheromone Trails:

Maintained for exam-timeslot-room combinations and updated based on solution quality and evaporation.

Heuristic Function:

Inverse of penalty score used to guide ants toward better paths.

Solution Construction:

Each ant:

- Estimates required timeslot count based on exam duration.
- Finds valid combinations of timeslots and rooms.
- Picks assignment using a probability function combining pheromone and heuristic.

Local Search Optimization:

Post-construction local search attempts:

- Exam swapping
- Room reallocation
- Timeslot shifting

Solution Representation

1. Encoded Format (String-Based)

This is a compact, string-based representation primarily used for:

- Genetic algorithm operations (crossover, mutation)
- ACO solution construction
- Efficient storage and manipulation

The encoded format follows this pattern:

"C1-TS1+TS2-R1+R2"

Where:

- **C1**: Course/Exam ID (e.g., "C1" for Course/Exam 1)
- **TS1+TS2**: Timeslot assignments, with "+" indicating multiple consecutive slots needed for longer exams
- **R1+R2**: Room assignments, with "+" indicating multiple rooms needed for larger student groups

A complete timetable solution is represented as an array of these encoded strings:

["C1-TS3-R1", "C2-TS1+TS2-R2", "C3-TS5-R3+R4", ...]

Advantages of Encoded Format:

- Compact representation for algorithms to process
- Easy to swap components during crossover operations
- Simple to modify during mutation operations
- Consistent format for pheromone tracking in ACO

2. Decoded Format (Object-Based)

object-based representation used for:

- Fitness evaluation
- Constraint checking
- Solution analysis and reporting

The decoded format uses actual object references:

```
{
    'exam': Exam_object,
    'timeslots': [Timeslot_objects],
    'rooms': [Room_objects]
}
```

Advantages of Decoded Format:

- Direct access to object properties (like room capacity, student lists)
- Easier constraint checking (student conflicts, room capacity)
- More intuitive for generating reports and visualizations

Fitness Function(Heuristic)

A penalty-based fitness function evaluates each generated timetable by combining several constraints using weighted scores:

Constraint	Weight (Penalty)
Room capacity exceeded	20
Room double-booking	40
Student conflict (same timeslot)	70
Multiple exams per day	25
Non-consecutive timeslots	50
Difficulty clustering	3
Weekend scheduling	10
Well-spread exam bonuses	-5

The total fitness is calculated as:

python

```
fitness = -(capacity_penalty + conflict_penalty + room_penalty + ... )
+ bonus
```

Conflict Detection and Management

Common conflict types tracked by both algorithms:

- **Student Conflicts**: Same student in multiple exams at once.
- **Room Conflicts**: Double-booked rooms.
- **Capacity Violations**: Overcrowded exam rooms.
- **Consecutive Exams**: Multiple exams for a student in a single day.
- **Non-Consecutive Slots**: Exams requiring multiple slots are scheduled non-consecutively.
- **Weekend Exams**: Scheduling on Fridays/Saturdays is penalized.

Timeslot and Room Management

- Ensures required consecutive slots are allocated for long-duration exams.
- Allocates multiple rooms when capacity overflows.
- Avoids weekends unless necessary.
- Preserves slot continuity and space efficiency.

Comparison between the 2 algorithms

Execution Time

- **GA (Genetic Algorithm): 64.13 seconds**
- **ACO (Ant Colony Optimization): 3521.42 seconds**
- **Conclusion**: GA is **~55x faster** than ACO. It is significantly more time-efficient and suitable for real-time or time-sensitive exam timetabling.

Solution Quality (Fitness)

- **GA Best Fitness: 27.0**
- **ACO Best Fitness: 40**
- **Note**: Lower fitness values indicate better solutions (fewer penalties).
- **Conclusion**: **GA produced a better-quality timetable** with fewer penalty points. ACO's solution, though valid, had a **higher penalty**.

Algorithm Convergence

- **GA**: Improved gradually over **42 generations**, showing a clear evolutionary process.
- **ACO**: Reached best fitness **in the first iteration** and then degraded to 40 in the second—this suggests poor convergence behavior in your current setup (e.g., too few iterations or improper pheromone update strategy).

Conflict Handling

Both algorithms **completely avoided hard constraints**:

- Student Conflicts: ✅ 0
- Room Conflicts: ✅ 0
- Capacity Issues: ✅ 0
- Multiple Exams/Day: ✅ 0
- Non-Consecutive Slots: ✅ 0

Both ACO and GA generate **valid conflict-free solutions**.

Final Recommendation

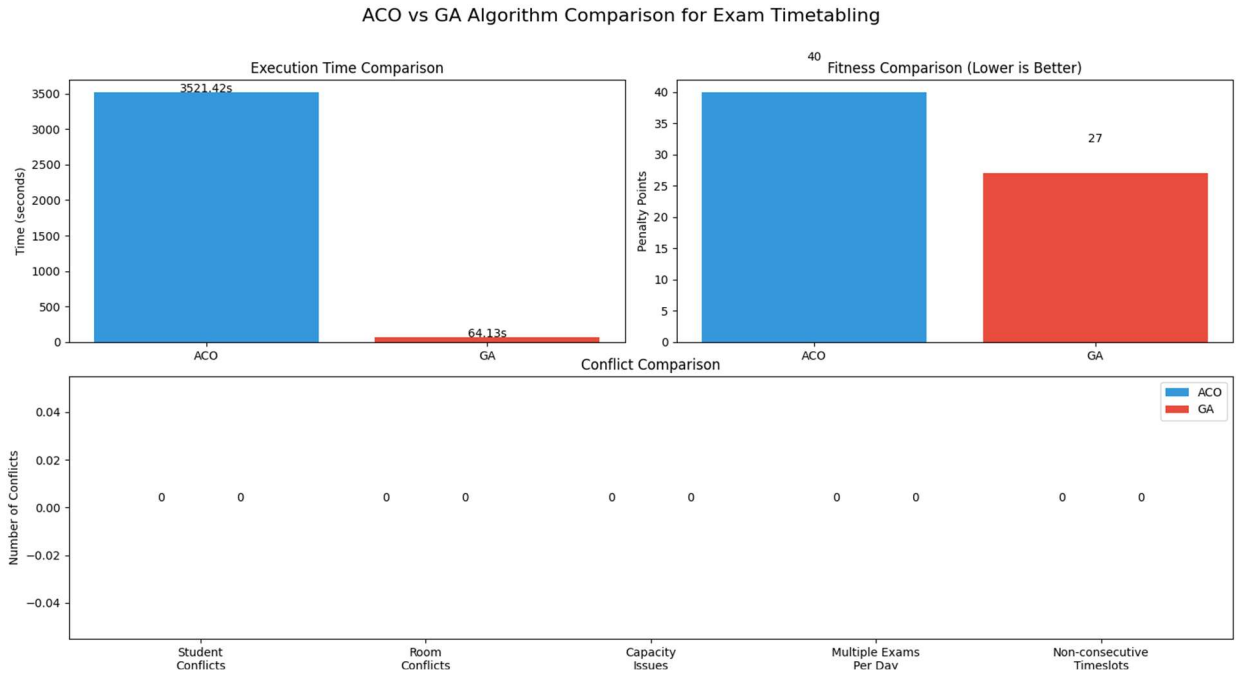
Metric	GA	ACO	Winner
Execution Time	✅ 64.13s	❌ 3521.42s	GA
Fitness (Lower Better)	✅ 27.0	❌ 40	GA
Conflicts	✅ 0	✅ 0	Tie
Stability/Convergence	✅ Steady progress	❌ Degraded result	GA

Final Verdict:

Genetic Algorithm (GA) is **superior** in both execution efficiency and solution quality for this exam timetabling problem.

Output and Visualization

- The final output includes a PDF-formatted timetable.
- Conflict reports highlight violations and suggest improvements.
- Optional console-based visualization groups exams by days and rooms.



Tools and Techniques Used

Python Libraries

1. reportlab
    - Modules: reportlab.lib.colors, reportlab.lib.pagesizes, reportlab.platypus, reportlab.lib.styles, reportlab.lib.units, reportlab.lib.enums
    - Used for generating PDF reports, including tables, paragraphs, styled content, and document layout.
2. matplotlib.pyplot
    - Used for creating visualizations such as fitness progression curves and conflict distribution charts.
3. numpy
    - Used for numerical computations, matrix operations, and efficient handling of large datasets.
4. collections.defaultdict
    - Utilized to manage grouped data structures such as students' enrolled exams and conflict mappings.
5. random
    - Used for probabilistic selection, mutation, and stochastic behavior in genetic and ACO algorithms.
6. datetime
    - Used for date and time handling in report timestamps and scheduling logic.
7. os
    - Used for file system operations, including saving and retrieving generated outputs.

Thank you